Sparse Approximation via Generating Point Sets

AVRIM BLUM, Toyota Technological Institute at Chicago
SARIEL HAR-PELED, University of Illinois, Urbana–Champaign
BENJAMIN RAICHEL, University of Texas at Dallas

For a set P of n points in the unit ball $\mathrm{b} \subseteq \mathbb{R}^d$, consider the problem of finding a small subset $T \subseteq P$ such that its convex-hull ε-approximates the convex-hull of the original set. Specifically, the Hausdorff distance between the convex hull of T and the convex hull of P should be at most ε. We present an efficient algorithm to compute such an ε'-approximation of size k_{alg}, where ε' is a function of ε and k_{alg} is a function of the minimum size k_{opt} of such an ε-approximation. Surprisingly, there is no dependence on the dimension d in either of the bounds. Furthermore, every point of P can be ε-approximated by a convex-combination of points of T that is $O(1/\varepsilon^2)$-sparse.

Our result can be viewed as a method for sparse, convex autoencoding: approximately representing the data in a compact way using sparse combinations of a small subset T of the original data. The new algorithm can be kernelized, and it preserves sparsity in the original input.

CCS Concepts: • **Theory of computation** → *Approximation algorithms analysis*; • **Computing methodologies** → *Unsupervised learning*;

Additional Key Words and Phrases: Convex hull, coreset, sparse approximation

ACM Reference format:
Avrim Blum, Sariel Har-Peled, and Benjamin Raichel. 2019. Sparse Approximation via Generating Point Sets. *ACM Trans. Algorithms* 15, 3, Article 32 (June 2019), 16 pages.
https://doi.org/10.1145/3302249

1 INTRODUCTION

This article deals with the topic of (doubly) sparse data representation. Namely, given a point set, we would like to identify a small subset such that each input point can be represented by a small combination from this subset. Such representations enable one to efficiently store and manipulate the data and can be used to expose the low dimensionality of the input. The desired sparsity is

Work by A.B. was conducted while on sabbatical at the University of Illinois and was partially supported by NSF awards CCF-1415460 and CCF-1525971. Work by S.H. was partially supported by NSF AF awards CCF-1421231 and CCF-1217462. Work by B.R. was partially supported by the University of Illinois Graduate College Dissertation Completion Fellowship, NSF CRII Award 1566137, and CAREER Award 1750780.
A preliminary version of this article appeared in SODA 16 [7]. The full version of the article is also available on the arxiv [6].
Authors' addresses: A. Blum, Toyota Technological Institute at Chicago (TTIC), 6045 S. Kenwood Ave. Chicago, IL 60637, USA; S. Har-Peled, SC 3306, Computer Science, UIUC, 201 N. Goodwin Avenue, Urbana, IL 61801, USA; B. Raichel, Department of Computer Science, Univeristy of Texas at Dallas, 800 W. Campbell Rd., MS EC-31, Richardson, TX 75080, USA.

1549-6325/2019/06-ART32 $15.00
https://doi.org/10.1145/3302249

twofold, both in the size of the generating set and the number of elements of this set used in representing each data point.

The Dimension of the Data. In the worst case, the data might be inherently high dimensional—making it hard to handle it efficiently. However, for many real-wold inputs, the data are close to being low dimensional in nature—either because they (almost) lie on a low-dimensional manifold or because they are highly clustered. The challenge as such is to use the inherent low-dimensional nature of the data (if it exists) to get more efficient algorithms.

Sparse Approximation and Coresets. Let P be a set of n points (observations) in the unit ball $\mathsf{b} \subseteq \mathbb{R}^d$, and let $\bigcirc P$ denote the convex-hull of P. Consider the problem of finding a small ε-coreset $T \subseteq P$ for projection width. Formally, given any line ℓ in $\mathbb{R}^d$ consider the projection of $\bigcirc T$ (respectively, $\bigcirc P$) onto the line ℓ—this results in an interval $I_T \subseteq \ell$ (respectively, $I_P \subseteq \ell$). We have that $I_T \subseteq I_P$. We require that $I_P \subseteq (1+\varepsilon) I_T$. Such coresets have size $O(1/\varepsilon^{(d-1)/2})$ and lead to numerous efficient approximation algorithms in low dimensions, see References [1]. In particular, such an ε-coreset guarantees that the Hausdorff distance between $\bigcirc T$ and $\bigcirc P$ is at most ε.

While such coresets can have size $\Omega(1/\varepsilon^{(d-1)/2})$ in the worst case, the dataset may have structure allowing much smaller coresets to exist even in high dimensional spaces. For example, consider a dataset P in which all points are ε-close to one of k different lines. Then taking the extreme dataset points associated with each line results in $2k$ points, such that every $p \in P$ is 2ε-close to the convex hull of those points (this is the merge-and-reduce property of coresets; see Reference [2, Section 5]). More generally, the union of any two datasets that have ε-close approximations of sizes k and k', respectively, has one of size at most $k + k'$. Thus, it is natural to ask whether one can approximate the smallest such coreset, in terms of both its size and approximation quality.

The Problem in Matrix Form. Given a collection P of n points (observations) in the unit ball $\mathsf{b} \subseteq \mathbb{R}^d$, viewed as column vectors, find a $d \times k$ matrix M such that each $p \in P$ can be approximately reconstructed as a *sparse, convex combination* of the columns of M. That is, for each $p \in P$ there exists a sparse non-negative vector x whose entries sum to one such that $p \approx Mx$. This problem is trivial if we allow $k = n$: Simply make each data point $p \in P$ into a column of M, allowing the ith data point to be perfectly reconstructed using $x = e_i$, where e_i is the ith vector in the standard basis. The goal is to do so using $k \ll n$, so that M and the x's can be viewed as an (approximate) compressed representation of the p's.

Input Assumption. We are given a set P of n points in $\mathbb{R}^d$ all with norm at most 1. Suppose that there *exists* a $d \times k_{\text{opt}}$ matrix M, such that

(A) each column of M is a convex combination of the observations p, and
(B) each $p \in P$ can be ε-approximately reconstructed as a convex combination of the columns of M: That is, for each $p \in P$ there exists a non-negative vector x whose entries sum to 1 such that $\|p - Mx\| \leq \varepsilon$.

Stated geometrically, the assumption is that the input P is contained in the unit ball b (centered at the origin), and there exists a set $P_{\text{opt}} \subseteq \bigcirc P$, of size k_{opt}, such that for any point $p \in P$, we have that p is ε-close to $\bigcirc P_{\text{opt}}$, where $\bigcirc P_{\text{opt}}$ denotes the convex-hull of P_{opt}. Formally, being *ε-close* means that the distance of p to the set $\bigcirc P_{\text{opt}}$ is at most ε.

Our Results. We present efficient algorithms for computing a $d \times k_{\text{alg}}$ matrix M', consisting of k_{alg} points of P, such that each $p \in P$ can be ε'-approximately reconstructed as a sparse convex combination of the columns of M', where k_{alg} and ε' are not too large, see Figure 1 for details. Here *sparse*

Technique	$\varepsilon' \equiv d_H(\bigcirc P, \bigcirc T)$	$k_{\text{alg}} \equiv \lvert T \rvert$	Result
ε-nets	$\leq \varepsilon$	$O(dk_{\text{opt}} \log k_{\text{opt}})$	Lemma 3.2
Greedy set cover	$\leq (1+\delta)\varepsilon$	$O\big(\big(k_{\text{opt}}/(\varepsilon\delta)^2\big) \log n\big)$	Lemma 3.3
Greedy clustering	$\leq 8\varepsilon^{1/3} + \varepsilon$	$O\big(k_{\text{opt}}/\varepsilon^{2/3}\big)$	Theorem 4.6 No dependency on d or n

Fig. 1. Summary of our results: Given a set P contained in the unit ball of $\mathbb{R}^d$, such that there is a subset $P_{\text{opt}} \subseteq P$ of size k_{opt}, and $d_H(\bigcirc P, \bigcirc P_{\text{opt}}) \leq \varepsilon$, the above results compute an approximate set $T \subseteq P$. Note that any point in P has an $O(1/\varepsilon^2)$-sparse $(\varepsilon + \varepsilon')$-approximation using T, because of the underlying sparsity—see Lemma 2.7.

means that only relatively few of the columns of M' would be used to represent (approximately) each point of *data*.

Stated in geometric terms, the algorithm computes a set T of k_{alg} points (these will be points from P) such that every point in P is ε'-close to the convex hull of T and, moreover, can be approximately reconstructed using a sparse convex combination of T.

The reader may notice that sparsity is not mentioned in the assumption about P_{opt} ($\equiv M$) and yet appears in the conclusion about T ($\equiv M'$). This is because convex combinations have the property that sparsity can be achieved almost for free, at the expense of a small amount of reconstruction error (see Lemma 2.7). This is to some extent the same reason that a large margin separator can be represented using a small number of support vectors.

Related Work. In comparison with the recent provable algorithms for autoencoding of Arora et al. [3], our result does not require any distributional assumptions on the x's or p's, e.g., that the $p \in P$ were produced by choosing x from a particular distribution and then computing Mx and adding random noise. It also does not require that the columns of M be incoherent (nearly orthogonal). However, we do require that the columns of M be convex combinations of the points $p \in P$ and that they can approximately reconstruct the $p \in P$ via convex combinations, so our results are incomparable to those of Arora et al. [3]. Work on related encoding or dictionary learning problems in the full rank case has been done by Spielman et al. [19], and efficient algorithms for finding minimal and sparse Boolean representations under anchor-set assumptions were given by Balcan et al. [4].

By considering all conic rather than convex combinations of the points, i.e., positive combinations that no longer need to sum to 1, our problem can be seen to be related to (a discrete version of) non-negative matrix factorization. Recently, Van Buskirk et al. [8] applied our results to the conic case by reducing it to the convex case when the angular spread is bounded. Moreover, they showed that given n points in $\mathbb{R}^{d+2}$, it is d-SUM-Hard to determine whether there is a subset of k points that ε-approximate the convex (or conic) hull, thus justifying the need to consider approximation algorithms in the current article.

1.1 The Results in Detail

Our results are summarized in Figure 1.

Preliminaries:

(A) **Sparse nearest-neighbor in high dimensions.** For a set of points P in the unit ball $\mathrm{b} \subseteq \mathbb{R}^d$ and any point $p \in \bigcirc P$, one can find a point $p' \in \bigcirc P$ that is the convex combination of $O(1/\varepsilon^2)$ points of P, such that $\|p - p'\| \leq \varepsilon$. This is, of course, well known by now [10], and we describe (for the sake of completeness) the surprisingly simple iterative

algorithm (which is similar to the Perceptron algorithm) to compute such a representation in Section 2.2. This sparse representation is sometimes referred to as an approximate Carathéodory theorem [5], and it also follows from the analysis of the Perceptron algorithm [18]—see Remark 2.8.

(B) **Geometric hitting set.** Our problem can be interpreted as (a somewhat convoluted) geometric hitting set problem. In particular, one can apply the Clarkson [9] polytope approximation algorithm to this problem, thus yielding an $O(d \log k_{\text{opt}})$ approximation in $n^{O(d)}$ time. For the sake of completeness, we describe this in detail in Section 3.1. (Since d might be large, this approximation is somewhat less attractive.)

Our results:

(C) **The greedy approach.** A natural approach is to try and solve the problem using the greedy algorithm. Here this requires some work, and the resulting algorithm is a combination of the algorithm from (A) with greedy set cover for the ranges defined in (B). We initialize an instance of the algorithm from (A) for each point $p \in P$ whose job is to either find a hyperplane through p separating it from $P \setminus \{p\}$ by a large margin or else to approximate p as a combination of a few support-vectors in $P \setminus \{p\}$. At each step, we find the point $p' \in P$ that causes as many of these algorithms to perform an update as possible and add it into our set T. The key issue is to prove that the procedure halts after a limited number of steps. This algorithm is described in Section 3.2.

(D) **Using greedy clustering.** The second algorithm, and our main contribution, is more similar in spirit to the Gonzalez algorithm for k-center clustering: Repeatedly find the point $p \in P$ that is farthest from the convex hull of the points of T and then add it into T if this distance is greater than some threshold (a similar idea was used for subspace approximation [14, Lemma 5.2]). The key issue here is to prove that some measure of significant progress is made each time a new point is added. Somewhat surprisingly, after $O(k_{\text{opt}}/\varepsilon^{2/3})$ iterations, the resulting set is an $O(\varepsilon^{1/3})$-approximation to the original set of points. Note, that unlike the other results mentioned above, there is no dependence on the dimension or the input size.

An additional property of all the above algorithms is that the points T found will be actual dataset points and the algorithms only require dot-product access to the data. This means that the algorithms can be kernelized.

Overview. We formally define the problem in Section 2.1. The workhorse of our algorithms is a procedure that computes the nearest point in a convex hull and is described in Section 2.2.

In Section 3, we present approximation algorithms that arise from the natural hitting set formulation. Specifically, in Section 3.1, we present a VC dimension-based rounding argument, with running time exponential in the dimension (i.e., $n^{O(d)}$). We present a greedy algorithm in Section 3.2—its running time is polynomial in n, but the constant term is subexponential in d. Both of these algorithms are presented as what can be done with known machinery.

Our main contribution is in Section 4, where we present a bi-criterion approximation algorithm for our problem that has polynomial running time in all parameters.

2 PRELIMINARIES

For a set $X \subseteq \mathbb{R}^d$, $\bigcirc X$ denotes the *convex hull* of X. For two sets $P, U \subseteq \mathbb{R}^d$, we denote by $d(P, U) = \min_{p \in P} \min_{u \in U} \|p - u\|$ the *distance* between P and P'. For a point $q \in \mathbb{R}^d$, its distance

to the set P is $d(q,P) = d(\{q\},P)$, and its **projection** or **nearest neighbor** in P is the point $\mathrm{nn}(q,P) = \arg\min_{p\in P} \|q-p\|$.

2.1 Sparse Convex-Approximation: Problem Statement and Background

For a set Y in $\mathbb{R}^d$, its *one-sided Hausdorff distance* from X is $d(Y \to X) = \max_{y\in Y} d(y,X)$.

Definition 2.1. Consider two sets $P_{\text{in}}, P_{\text{out}} \subseteq \mathbb{R}^d$. A set $U \subseteq \bigcirc P_{\text{out}}$ is a **δ-approximation** to P_{in} from P_{out} if $d(\bigcirc P_{\text{in}} \to \bigcirc U) \le \delta$. In words, every point of $\bigcirc P_{\text{in}}$ is within distance δ from a point of $\bigcirc U$. In the **discrete δ-approximation** version, we require that $U \subseteq P_{\text{out}}$. We use $\mathrm{opt}(P_{\text{in}}, P_{\text{out}}, \delta)$ to denote any minimum cardinality discrete δ-approximation to P_{in} from P_{out} and $k_{\text{opt}}(P_{\text{in}}, P_{\text{out}}, \delta) = |\mathrm{opt}(P_{\text{in}}, P_{\text{out}}, \delta)|$ to denote its size. We drop the phrase "from P_{out}" when it is clear from the context.

Problem 2.2. *Given sets $P_{\text{in}}, P_{\text{out}} \subseteq \mathbb{R}^d$, compute (or approximate)* $\mathrm{opt}(P_{\text{in}}, P_{\text{out}}, \delta)$.

For the majority of the article, we focus on the natural special case when $P = P_{\text{in}} = P_{\text{out}}$. The **Hausdorff distance** between sets X and Y is defined as $d_H(X,Y) = \max(d(Y \to X), d(X \to Y))$.

Lemma 2.3.

(i) For a convex set $C \subseteq \mathbb{R}^d$, the function $f(p) = d(p,C)$ is convex, where $p \in \mathbb{R}^d$.
(ii) A convex-function f, over a convex bounded domain $D \subseteq \mathbb{R}^d$, attains its maximum in a boundary point of D.
(iii) For bounded point sets $U, P \subseteq \mathbb{R}^d$, such that $U \subseteq \bigcirc P$, we have $d_H(\bigcirc U, \bigcirc P) = d(P \to \bigcirc U)$.

Proof. This is all well known, and we include the proof for the sake of completeness.

(i) Consider any two points p, y in $\mathbb{R}^d$, and let $p' = \mathrm{nn}(p,C)$ and $y' = \mathrm{nn}(y,C)$. For any $t \in [0,1]$, we have by convexity that $z = tp + (1-t)y \in py$ (i.e., z is a convex combination of p and y) and $z' = tp' + (1-t)y' \in C$. Therefore, by the triangle inequality, we have

$$\begin{aligned} f(z) = f\left(tp + (1-t)y\right) &\le \|z - z'\| = \left\|\left(tp + (1-t)y\right) - \left(tp' + (1-t)y'\right)\right\| \\ &= \left\|t(p-p') + (1-t)(y-y')\right\| \le \left\|t(p-p')\right\| + \left\|(1-t)(y-y')\right\| \\ &= t\left\|p-p'\right\| + (1-t)\left\|y-y'\right\| = tf(p) + (1-t)f(y). \end{aligned}$$

(ii) If p is the interior of D, then there are extremal points $p_1, \ldots, p_d$ of D, and constants $\alpha_1, \ldots, \alpha_d \in [0,1]$, such that $\sum_i \alpha_i = 1$ and $p = \sum_i \alpha_i p_i$. As such, by convexity, we have $f(p) = f(\sum_i \alpha_i p_i) \le \sum_i \alpha_i f(p_i) \le \max_i f(p_i)$.

(iii) By (i), the function $d(p, \bigcirc U)$ is convex. By (ii), its maximum over $\bigcirc P$ is attained at a point of P. We thus have that

$$\begin{aligned} d_H(\bigcirc U, \bigcirc P) &= \max\left(d\left(\bigcirc U \to \bigcirc P\right), d\left(\bigcirc P \to \bigcirc U\right)\right) = d(\bigcirc P \to \bigcirc U) = \max_{p \in \bigcirc P} d\left(p, \bigcirc U\right) \\ &= \max_{p \in P} d\left(p, \bigcirc U\right) = d\left(P \to \bigcirc U\right). \end{aligned}$$

□

Definition 2.4. Consider any set $P \subseteq \mathbb{R}^d$. A set $U \subseteq \bigcirc P$ is a **δ-approximation** to P if $d_H(\bigcirc U, \bigcirc P) \le \delta$. By the above lemma, this is equivalent to every point of P being in distance at most δ from a point of $\bigcirc U$. In the **discrete δ-approximation** version, we require that $U \subseteq P$. Let $\mathrm{opt}(P,\delta)$ be any minimum cardinality δ-approximation to P, and let $k_{\text{opt}}(P,\delta) = |\mathrm{opt}(P,\delta)|$ denote its size.

Problem 2.5. *Given a set $P \subseteq \mathbb{R}^d$ and value δ, compute (or approximate)* $\mathrm{opt}\,(P,\delta)$.

Example 2.6. Consider a unit radius sphere $\mathbb{S}^{(d-1)}$ in $\mathbb{R}^d$ centered at the origin, and let P be a δ'-packing on $\mathbb{S}^{(d-1)}$ (i.e., every point in $\mathbb{S}^{(d-1)}$ is at distance at most δ' from a point of P, and

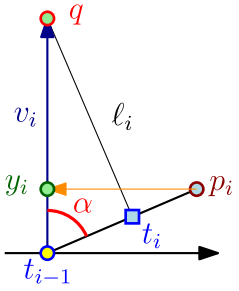


Fig. 2. The points and other quantities defined in Lemma 2.7.

any two points of P are at distance at least δ' from each other). It is easy to verify that such a δ'-packing has size $\Theta(1/(\delta')^{d-1})$. Furthermore, for any $\delta > 0$, and an appropriate absolute constant c (independent of the dimension or δ), setting $\delta' = c\sqrt{\delta}$, we have the property that for any point $p \in P$, $d(p, \bigcirc P \setminus \{p\}) > \delta$. That is, any δ-approximation to P requires $\Omega(1/\delta^{(d-1)/2})$ points.

However, let $P_{\text{out}} = \{\pm d\mathrm{e}_i \mid i = 1, \ldots, d\}$, where e_i denotes the ith orthonormal vector, having zero in all coordinates except for the ith coordinate, where it is 1. Clearly, $\mathbb{S}^{(d-1)} \subseteq \bigcirc P_{\text{out}}$, and as such $k_{\text{opt}}(P, P_{\text{out}}, \delta) \leq |P_{\text{out}}| = 2d$, with equality for $\delta = 0$.

Throughout this article, we require that P_{out} be contained in the unit ball, disallowing this latter type of "trivial" solution, and, furthermore, having the property that a successful approximation also yields a *sparse* solution essentially for free, as shown next in Lemma 2.7.

2.2 Algorithm for Approximately Computing the Distance to the Convex Hull

The following is well known and is included for the sake of completeness, see Reference [13]. It also follows readily from the Preceptron algorithm (see Remark 2.8 below).

Lemma 2.7. *Let $P \subseteq \mathbb{R}^d$ be a point set, $\varepsilon > 0$ be a parameter, and $q \in \mathbb{R}^d$ be a given query point. Then, one can compute, in $O(|P|\, d/\varepsilon^2)$ time, a point $t \in \bigcirc P$, such that $\|q - t\| \leq d(q, \bigcirc P) + \varepsilon\Delta$, where $\Delta = \operatorname{diam}(P)$. Furthermore, t is a convex combination of $O(1/\varepsilon^2)$ points of P.*

Proof. The algorithm is iterative, computing a sequence of points $t_0, \ldots, t_i$ inside $\bigcirc P$ that approach q. Initially, $p_0 = t_0$ is the closest point of P to q. In the ith iteration, the algorithm computes the vector $v_i = q - t_{i-1}$, and the point $p_i \in P$ that is extremal in the direction of v_i, see Figure 2. Now, the algorithm sets t_i to be the closest point to q on the segment $s_i = t_{i-1}p_i$, and continues to the next iteration, for $M = O(1/\varepsilon^2)$ iterations. The algorithm returns the point t_M as the desired answer.

By induction, the point $t_i \in \bigcirc\{p_0, \ldots, p_i\}$. Furthermore, observe that the distance of the points $t_0, t_1, \ldots$ from q is monotonically decreasing. In particular, for all $i > 0$, t_i must fall in the middle of the segment s_i, as otherwise p_i would be closer to q than p_0, a contradiction to the definition of p_0.

Project the point p_i to the segment $t_{i-1}q$, and let y_i be the projected point. Observe that $\|q - y_i\|$ is a lower bound on $d(q, \bigcirc P)$. Therefore, if $\|y_i - t_{i-1}\| \leq \varepsilon\Delta$, then we are done, as $\|q - t_{i-1}\| \leq \|t_{i-1} - y_i\| + \|y_i - q\| \leq \varepsilon\Delta + d(q, \bigcirc P)$. (In particular, one can use this as alternative stopping condition for the algorithm instead of counting iterations.)

So let α be the angle $\angle p_i t_{i-1} q$. Observe that as $t_{i-1}p_i \subseteq \bigcirc P$, it follows that $\|t_{i-1} - p_i\| \leq \operatorname{diam}(P) = \Delta$. Furthermore, $\cos\alpha = \frac{\|y_i - t_{i-1}\|}{\|t_{i-1} - p_i\|} > \frac{\varepsilon\Delta}{\Delta} = \varepsilon$, since $\|y_i - t_{i-1}\| > \varepsilon\Delta$. Hence, $\sin\alpha = \sqrt{1 - \cos^2\alpha} \leq \sqrt{1 - \varepsilon^2} \leq 1 - \varepsilon^2/2$. Let $\ell_{i-1} = \|q - t_{i-1}\|$. We have that

$$\ell_i = \|q - t_i\| = \|q - t_{i-1}\| \sin\alpha \leq (1 - \varepsilon^2/2)\ell_{i-1}.$$

Analyzing the number of iterations required by the algorithm is somewhat tedious. If $\ell_0 = \|q - t_0\| \geq (4/\varepsilon^2)\Delta$, then the algorithm would be done in one iteration as otherwise $\ell_1 \leq \ell_0 - 2\Delta$, which is impossible. In particular, after $4/\varepsilon^2$ iterations the distance ℓ_i shrinks by a factor of two, and, as such, after $O((1/\varepsilon^2)\log(1/\varepsilon))$ iterations the algorithm is done.

One can do somewhat better. By the above, we can assume that $d(q, P) = O(\Delta/\varepsilon^2)$. Now set $\varepsilon_j = 1/2^{2+j}$. By the above, after $n_0 = O((1/\varepsilon_0^2)\log(1/\varepsilon_0)) = O(1)$ iterations, $\ell_{n_0} \leq d(q, \bigcirc P) + \operatorname{diam}(P)/4$. For $j \geq 1$, let $n_j = 4/(\varepsilon_j)^2$, and observe that, after $\nu_j = n_j + \sum_{k=0}^{j-1} n_k$ iterations, we have that

$$\ell_{\nu_j} \leq \left(d(q, \bigcirc P) + \varepsilon_{j-1}\Delta\right)/2 \leq d(q, \bigcirc P) + \varepsilon_j\Delta.$$

In particular, stopping as soon as $\varepsilon_j \leq \varepsilon$, we have the desired guarantee, and the number of iterations needed is $M = O(1) + \sum_{j=0}^{\lceil \lg 1/\varepsilon \rceil} 4/\varepsilon_j^2 = O(1/\varepsilon^2)$. □

In our use of Lemma 2.7, P and q will always be contained in the unit ball, so we can remove the Δ term in the bound if we wish since $\Delta \leq 2$.

Remark 2.8. Lemma 2.7 is known, and a variant of it follows readily from a result (from 1962) on the convergence of the Perceptron algorithm [18]. Indeed, consider a set $P \subseteq \mathbb{R}^d$ and a query point $q \in \mathbb{R}^d$. Assume that $q \in \bigcirc P$ and, furthermore, that q is the origin (translating space if needed to ensure this). Run the Perceptron algorithm learning a linear classifier that passes through the origin and classifies P as positive examples. Stop the algorithm after $M = 1/\varepsilon^2$ classification mistakes (since $q \in \bigcirc P$, there will always be a mistake in P). Let $p_1, \ldots, p_M$ be the sequence of points on which mistakes were made, and let $w = p_1 + \cdots + p_M$ be the resulting hypothesis vector. By the analysis of Reference [18], we have $\|w\| \leq \operatorname{diam}(P)\sqrt{M}$. This implies that the point $p' = w/M$, which is a convex combination of the points $p_1, \ldots, p_M$, has length—and therefore distance from q—at most $\varepsilon\operatorname{diam}(P)$.

Thus, we conclude that for any point $p \in \bigcirc P$, and any $\varepsilon \in (0, 1)$, there is a point $p' \in \bigcirc U$ that is a convex combination of $O(1/\varepsilon^2)$ points of P, such that $\|p - p'\| \leq \varepsilon\operatorname{diam}(P)$. This is sometimes referred to as the approximate Carathéodory theorem, which has been shown to apply more generally to any ℓ_p norm, for $p \geq 2$, where the number of points output is $O(p/\varepsilon^2)$ [5, 17]. We described the alternative algorithm (in the proof of Lemma 2.7) for the Euclidean norm, because it is more direct and slightly simpler in this case.

Remark 2.9. The exact nearest-neighbor problem is somewhat more challenging. Specifically, given a point p, and a set of points U, both in $\mathbb{R}^d$, computing the nearest point to p in $\bigcirc U$ (i.e., $\operatorname{nn}(q, \bigcirc U)$) can be written as quadratic optimization problem. It can be solved using LP-type techniques [12, Section 15.5] in time roughly $2^{O(\sqrt{d \log d})} n$, where $n = |U|$. Alternatively, weakly polynomial time algorithms are known using the ellipsoid method. In the following, let $T_{\mathrm{nn}}(n)$ denote the running time of this procedure.

3 APPROXIMATION ALGORITHMS VIA HITTING SET FORMULATION

Here we look at two hitting set–type algorithms for Problem 2.2. An $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-**approximation** of $\operatorname{opt}(P_{\mathrm{in}}, P_{\mathrm{out}}, \varepsilon)$ is a set $U \subseteq P_{\mathrm{out}}$ such that $d(\bigcirc P_{\mathrm{in}} \to \bigcirc U) \leq \alpha$ and $|U| \leq \beta k_{\mathrm{opt}}(P_{\mathrm{in}}, P_{\mathrm{out}}, \varepsilon)$, see Definition 2.1.

As a warm-up exercise, we first present an $(\varepsilon, O(d \log k_{\mathrm{opt}}))$-approximation using approximation algorithms for hitting sets for set systems with bounded VC dimension. Then we build on that to get a greedy algorithm providing a $((1+\delta)\varepsilon, O((\varepsilon\delta)^{-2}\log n))$-approximation, where n is the total number of input points.

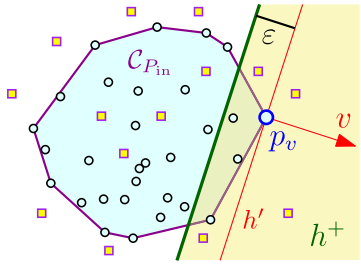
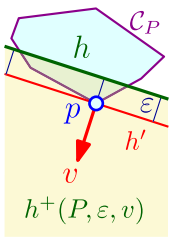


Fig. 3. Circles and squares denote points of P_{in} and P_{out}, respectively.

3.1 Approximation via VC Dimension

Definition 3.1. For a set $P \subseteq \mathbb{R}^d$ and a direction vector v, let p be the point of P extreme in the direction of v, and let h' be the hyperplane with normal v and tangent to $\bigcirc P$ at p. For a parameter ε, let h be the hyperplane formed by translating h' distance ε in the direction $-v$. The **ε-shadow** of h' (or v) is the halfspace $h^+(P, \varepsilon, v)$ bounded by h that contains p in its interior. In words, the ε-shadow of v is the outer supporting halfspace for P with a normal in the direction of v, translated in by distance ε.



Lemma 3.2. *Given sets P_{in} and P_{out} in $\mathbb{R}^d$ with a total of n points, and a parameter ε, one can compute a $(\varepsilon, O(d \log k_{\text{opt}}))$-approximation to the optimal discrete set* opt $(P_{\text{in}}, P_{\text{out}}, \varepsilon)$ *in $n^{O(d)}$ time.*

Proof. For a direction v, consider the hyperplane h' tangent to $\bigcirc P_{\text{in}}$ at an extremal point $p_v \in P_{\text{in}}$ in the direction of v and its ε-shadow $h^+ = h^+(P_{\text{in}}, \varepsilon, v)$, see Figure 3.

Clearly, any discrete ε-approximation $U \subseteq P_{\text{out}}$ to P_{in} must contain at least one point of $P_{\text{out}} \cap h^+$, as otherwise the approximation fails for the point p_v (in particular, if such a halfspace has no point in P_{out}, then there is no approximation). Now, consider the set system

$$\mathcal{S} = (P_{\text{out}}, \{P_{\text{out}} \cap h^+(P_{\text{in}}, \varepsilon, v) \mid v \text{ any unit vector}\}).$$

This set system has VC dimension at most $d + 1$, and, in particular, for such a set system one can compute a $O(d \log k_{\text{opt}})$ approximation to its minimum size hitting set, which is the desired approximation in this case, see Reference [12, Section 6.3]. We describe the algorithm below, but first we verify that this indeed yields the desired approximation.

Consider a hitting set $U \subseteq P_{\text{out}}$ of $\mathcal{S}$. Let p be any point in $\bigcirc P_{\text{in}}$, and let p' be the closest point to p in $\bigcirc U$. If $\|p - p'\| \leq \varepsilon$, then we are done. Otherwise, consider the vector $v = p - p'$. Let z denote the hyperplane whose normal is v and that passes through the point p', and let z^+ denote the open halfspace bounded by z and in the direction of v (i.e., containing p). As p' is the closest point to p in $\bigcirc U$, z^+ has empty intersection with $\bigcirc U$. Moreover, $h^+(P_{\text{in}}, \varepsilon, v) \subsetneq z^+$, as the bounding hyperplanes of both halfspaces have v as a normal, and the extreme point of $\bigcirc P_{\text{in}}$ in the direction of v must be $> \varepsilon$ away from z (as p is at least this far in the direction of v). See Figure 4. These two facts combined imply $h^+(P_{\text{in}}, \varepsilon, v) \cap \bigcirc U = \emptyset$, a contradiction as $h^+(P_{\text{in}}, \varepsilon, v) \cap P_{\text{out}}$ is a set in $\mathcal{S}$ that should have been hit.

As for the algorithm, Clarkson [9] described how to compute this set via reweighting, but the following technique due to Long [16] is easier to describe (we sketch it here for the sake of completeness). Consider the LP relaxation of the hitting set for this set system. Clearly, one can assign weights to points (between 0 and 1), such that the total weight of the points is at most k_{opt}, and for every range in $\mathcal{S}$ the total weight of the points it covers is at least 1. Dividing this fractional solution by k_{opt}, we get a weighted set system, where every set has weight at least $\eta = 1/k_{\text{opt}}$, and total

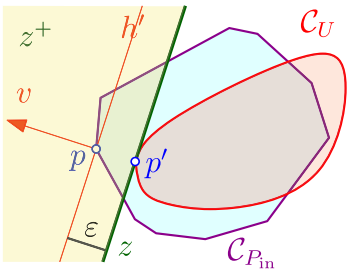


Fig. 4. The various quantities defined in Lemma 3.2.

weight of the points is 1. That is, we can interpret these weights over the points as a measure, where all the sets of interests are η-heavy. A random sample of size $O((d/\eta)\log(1/\eta)) = O(k_{\text{opt}} d \log k_{\text{opt}})$ of P (according to the weights) is an η-net with constant probability [15] and stabs all the sets of $\mathcal{S}$, as desired. Should the random sample fail, one can sample again until success. □

Note that in the above algorithm, the ranges in set system $\mathcal{S}$ can be enumerated in $n^{O(d)}$ time, since the boundary of each ε-shadow is a hyperplane. That is, it suffices to consider the set of all canonical hyperplanes with d input points on their boundary.

3.2 Approximation via a Greedy Algorithm

Lemma 3.3. *Let P_{in} and P_{out} be sets of points in $\mathbb{R}^d$ contained in the unit ball, with a total of n points. For parameters $\varepsilon, \delta \in (0,1)$, and $\tau = O(\varepsilon^{-2}\delta^{-2}\log n)$, one can compute a $((1+\delta)\varepsilon, \tau)$-approximation to the optimal discrete set $\text{opt}(P_{\text{in}}, P_{\text{out}}, \varepsilon)$. Namely, the algorithm outputs a subset of points of size $K = k_{\text{opt}}\tau = O(k_{\text{opt}}\varepsilon^{-2}\delta^{-2}\log n)$. The running time of the algorithm is $O((T_{\text{nn}}(K)+n)nK)$, where $T_{\text{nn}}(K)$ is the time to compute the nearest-neighbor to a convex-hull of a set of K points in $\mathbb{R}^d$, see Remark 2.9.*

Proof. The algorithm is greedy—the basic idea is to restrict the set system of Lemma 3.2 to the relevant active sets. Formally, let $U_0 = \{p_0\}$, where p_0 is some arbitrary point of P_{out}. For $i > 0$, in the ith iteration, consider the current convex set $C_{i-1} = \bigcirc U_{i-1}$. For a point $q \in P_{\text{in}} \setminus C_{i-1}$, let $\text{nn}(q, C_{i-1})$ be its nearest point in C_{i-1}, and let $v_i(q)$ be the direction of the vector $q - \text{nn}(p, C_{i-1})$. In particular, consider the ε-shadow halfspace $h^+ = h^+(P_{\text{in}}, \varepsilon, v_i(q))$, see Definition 3.1, which should be hit by the desired hitting set.[1]

Let $Z_i \subseteq P_{\text{in}}$ be the set of points of P_{in} that are *unhappy*; that is, they are in distance $\geq (1+\delta)\varepsilon$ from $\bigcirc U_{i-1}$. We restrict our attention to the set system of active halfspaces; that is,

$$\mathcal{S}_i = \left(P_{\text{out}}, \left\{P_{\text{out}} \cap h^+\left(P_{\text{in}}, \varepsilon, v_i(q)\right) \;\middle|\; q \in Z_i\right\}\right).$$

(As before, if $P_{\text{out}} \cap h^+$ is empty, then no approximation is possible, and the algorithm is done.) Now, as in the classical algorithm for hitting set (or set cover), pick the point p_i in P_{out} that hits the largest number of ranges in $\mathcal{S}_i$ and add it to U_{i-1} to form U_i.

A point $q \in Z_i$ is *hit* in the ith iteration if $p_i \in h^+(P, \varepsilon, v_i(q))$, see Figure 5. The argument of Lemma 2.7 (or Remark 2.8) implies that after a point $q \in P_{\text{in}}$ is hit $c/(\varepsilon^2\delta^2)$ times, its distance to the convex-hull of the current points is smaller than $(1+\delta)\varepsilon$, and it is no longer unhappy, where c is some sufficiently large constant. Indeed, using the notation of the proof Lemma 2.7, if a point $q \in Z_i$ is hit in the ith iteration by a point p_i, and $d(q, \bigcirc U_{i-1}) \leq (1+\delta)\varepsilon$, then we are done. Otherwise, let $t_{i-1} = \text{nn}(q, \bigcirc U_{i-1})$, and let y_i be the projection of p_i to the segment qt_{i-1}, see Figure 2. We have

[1] The hitting set computed by the algorithm is somewhat weaker, only hitting all the $(1+\delta)\varepsilon$-shadows.

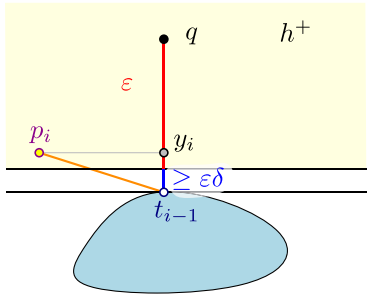


Fig. 5. Point q is hit in the ith iteration when $p_i \in h^+\big(P, \varepsilon, v_i(q)\big)$.

that $\|y_i - t_{i-1}\| \geq \|q - t_{i-1}\| - \|q - y_i\| \geq (1+\delta)\varepsilon - \varepsilon \geq \varepsilon\delta$, since $\|q - y_i\| \leq \varepsilon$ (as p_i and y_i are both in the ε-shadow of q). Now the analysis of Lemma 2.7 applies (with $\varepsilon\delta$ instead of ε), implying that after $O(1/(\varepsilon\delta)^2)$ iterations, the distance of q from the current convex-hull would be smaller than $(1+\delta)\varepsilon$.

So let n_i be the number of unhappy points in the beginning of the ith iteration, and observe that at least n_i/k_{opt} points are being hit in the ith iteration. In particular, let $\kappa = 2\lceil ck_{\text{opt}}/(\varepsilon^2\delta^2)\rceil$, and observe that in the iterations between $i-\kappa$ and i, we have that the number of points being hit is at least $\sum_{j=i-\kappa}^{i} n_j/k_{\text{opt}} \geq 2n_i c/(\varepsilon^2\delta^2)$. This implies that $n_{i-\kappa} \geq 2n_i$. Otherwise, $n_{i-\kappa} < 2n_i$, implying that in this range of iterations $> N = n_{i-\kappa}c/(\varepsilon^2\delta^2)$ hits happened, which is impossible, as $n_{i-\kappa}$ points can be hit at most N times before they are all happy.

As such, after κ iterations of the greedy algorithm, the number of unhappy points drops by a factor of two, and we conclude that after $O(k_{\text{opt}}\,(\varepsilon\delta)^{-2}\log n)$ total iterations, the algorithm is done.

As for the running time, observe that the algorithm needs to maintain for each point of P its nearest neighbor in the current convex-hull. As such, each iteration requires n computations of nearest-neighbor, which can be done in $T_{\text{nn}}(K)$ time, see Remark 2.9. □

4 APPROXIMATING THE CONVEX HULL IN HIGH DIMENSIONS

Here we provide an efficient bi-criteria approximation algorithm for Problem 2.5. That is, the algorithm computes a subset $U \subseteq \bigcirc P$, such that (i) $d_H(\bigcirc U, \bigcirc P) \leq O(\varepsilon^{1/3})\ \text{diam}(P)$ and (ii) $|U| \leq O(k_{\text{opt}}(P,\varepsilon)/\varepsilon^{2/3})$. Significantly, the computed set U is actually a subset of P, implying that the algorithm simultaneously solves both the continuous and discrete variants of the problem.

To simplify the presentation, in the remainder of this section we assume $\Delta = \text{diam}(P) = O(1)$ and hence drop most appearances of Δ.

4.1 The Algorithm

Let $\delta = 8\varepsilon^{1/3}$. The algorithm is greedy, similar in spirit to the Gonzalez algorithm for k-center clustering [11] and subspace approximation algorithms [14, Lemma 5.2]. The algorithm starts with an arbitrary point $t_0 \in P$. For $i > 0$, in the ith iteration, the algorithm computes the point t_i in P that is furthest away from $\bigcirc U_{i-1}$, where $U_{i-1} = \{t_0, \ldots, t_{i-1}\}$. For now, assume these distance queries are done exactly—later we describe how to use approximate queries (i.e., Lemma 2.7). Let $r_i = d(t_i, \bigcirc U_{i-1})$. The algorithm stops as soon as $r_i \leq \delta$ and outputs U_{i-1}.

Observation 4.1. *In the above algorithm, for all $i > 0$, the point t_i is a vertex of $\bigcirc P$ (so long as exact distance queries are used). In particular, if the output has to be a subset of the convex hull vertices, then one can choose t_0 to be the extreme vertex in any direction.*

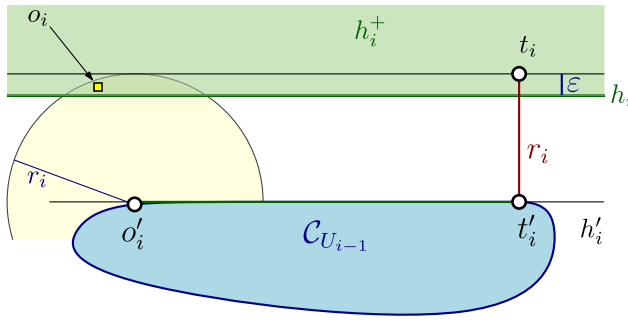


Fig. 6. Setup for Claim 4.2, bounding $\|o_i - o_i'\|$.

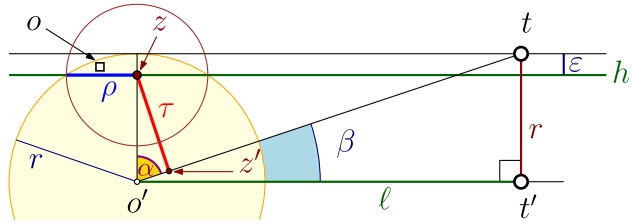


Fig. 7. The various defined quantities in Lemma 4.3.

4.2 Analysis

By the termination condition of the algorithm, when the algorithm stops every point in P is in distance at most $\delta = 8\varepsilon^{1/3}$ away from $\bigcirc U_{i-1}$, as desired. As for the number of rounds until termination, we argue that in each round there exists some point $o \in P_{\text{opt}}$ that is far from $\bigcirc U_{i-1}$ (as specified in Claim 4.2) and such that $d(o, U_i) \leq (1 - \Omega(\varepsilon^{2/3}))d(o, U_{i-1})$.

So consider some round i, the current set U_{i-1}, and the point $t_i \in P$ furthest away from $\bigcirc U_{i-1}$. Let t_i' be the closest point to t_i in $\bigcirc U_{i-1}$, and let $r_i = \|t_i - t_i'\|$. Let h_i be the hyperplane orthogonal to the segment $t_i t_i'$ and lying ε distance below t_i in the direction of t_i'. Let h_i^+ denote the closed halfspace having h_i as its boundary and that contains t_i, see Figure 6. If no points of P_{opt} are in h_i^+, then $d(t_i, \bigcirc P_{\text{opt}}) > \varepsilon$, which is impossible. Therefore, there must be a point $o_i \in P_{\text{opt}} \cap h_i^+$. Let o_i' be the closest point to o_i in $\bigcirc U_{i-1}$.

Claim 4.2. $r_i - \varepsilon \leq \|o_i - o_i'\| \leq r_i$.

Proof. Let h_i' be the translation of h_i so it passes through t_i', see Figure 6. We have that $r_i - \varepsilon = d(h_i', h_i) \leq \|o_i - o_i'\|$, as o_i lies in h_i^+ (i.e., above h_i) and all of $\bigcirc U_{i-1}$ lies below h_i'.

For the second part, for any $p \in \mathbb{R}^d$, let $f_{i-1}(p)$ be the distance of p from $\bigcirc U_{i-1}$. By Lemma 2.3 (iii), and since $o_i \in P_{\text{opt}} \subseteq \bigcirc P$, it follows that $\|o_i - o_i'\| \leq \max_{p \in \bigcirc P} f_{i-1}(p) = \|t_i - t_i'\| = r_i$. □

Lemma 4.3. *If $r_i \geq 8\varepsilon^{1/3}$, then $d(o_i, \bigcirc U_i) \leq (1 - \varepsilon^{2/3}) d(o_i, \bigcirc U_{i-1})$.*

Proof. In the following, all entities are defined in the context of the ith iteration, and we omit the subscript i denoting this to simplify the exposition. Assume, for the time being, that the angle $\angle tt'o'$ is a right angle and the segment $t'o'$ has length $\ell = 1$, see Figure 7. This is the worst-case configuration in terms of the new convex-hull $\bigcirc U_i$ getting closer to o, as can be easily seen.

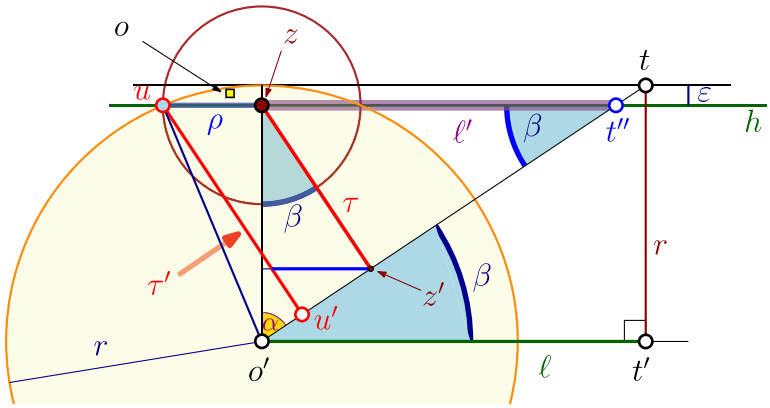


Fig. 8. Note that o is not necessarily in the two-dimensional plane depicted by the figure. All other points are in this plane.

(Specifically, any other configuration would cause the angle $\alpha = \angle zo't$ in Figure 7 to decrease, which below we argue needs to be upper bounded.)

Let z be the intersection of h with the ray emanating from o' in the direction $t - t'$. Let z' be the closest point to z on $o't$, let $\tau = \|z - z'\|$, and let ρ be the radius of the ball formed by $\text{ball}(o', r) \cap h$. See Figure 7.

Rather than bounding the distance of o to $\bigcirc U_i$ directly, we instead use bounds on ρ and τ. Observe that $o \in h^+ \cap \text{ball}(o', r) \subseteq \text{ball}(z, \rho)$, and, as such, $\|o - z\| \leq \rho$. Now we have $\rho = \sqrt{r^2 - \|z - o'\|^2} = \sqrt{r^2 - (r - \varepsilon)^2} = \sqrt{2r\varepsilon - \varepsilon^2} \leq \sqrt{2r\varepsilon}$.

Let $\alpha = \angle zo't$ and $\beta = \pi/2 - \alpha = \angle to't'$, and observe that $\sin\alpha = \cos\beta = \ell/\sqrt{\ell^2 + r^2}$, where $\ell = \|o' - t'\| = 1$. Now we have

$$\frac{\tau}{r - \varepsilon} = \sin\alpha = \frac{\ell}{\sqrt{\ell^2 + r^2}} = \frac{1}{\sqrt{1 + r^2}} \leq \sqrt{1 - \frac{r^2}{2}} \leq 1 - \frac{r^2}{4}, \tag{4.1}$$

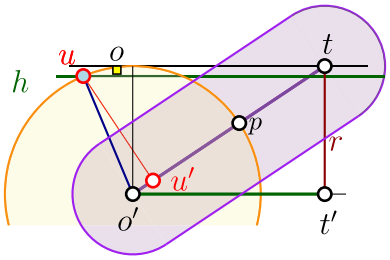
since $\ell = 1$ and $r \leq 1$.

Sanity Condition. Consider the line that is the intersection of the hyperplane h and the two-dimensional plane spanned by t, t', and o' (this line is denoted by h in the figures). Let u be the point in distance ρ on this line from z, on the side further away from t. Let t'' be the intersection of h with to'. Next, let u' be the nearest point to u on the segment to', see Figure 8.

We want to argue that the distance between o and $\bigcirc U_i$ can be bounded in terms of the distance between u and u'; however, to do so we need to guarantee that u' is in the interior of this segment to'. Setting $\ell' = \|z - t''\|$, this happens if

$$\begin{aligned}\|u' - t''\| < \|t'' - o'\| &\iff \|u' - t''\| = (\rho + \ell')\cos\beta = (\rho + \ell')\frac{\ell'}{\|t'' - o'\|} < \|t'' - o'\| \\ &\iff (\rho + \ell')\,\ell' < \|t'' - o'\|^2 = (\ell')^2 + (r - \varepsilon)^2\,.\end{aligned}$$

Thus, we have to prove that $\rho\ell' < (r - \varepsilon)^2$. As $\ell' < \ell = 1$, we have that this is implied if $\rho \leq \sqrt{2r\varepsilon} < (r - \varepsilon)^2$, and this inequality holds if $r \geq 8\varepsilon^{1/3}$.



Back to the proof: We next bound the distance of o from $\bigcirc U_i$. Observe that by rotating o around the line $o't$, we can assume that o lies on the plane spanned by t, t', o' and its distance to the segment to' has not changed. Now the set of points in distance r' from the segment $o't$ is a hippodrome, and this hippodrome covers a connected portion of $\text{ball}(o', r)$. For $r' = \|u - u'\|$, by the above sanity condition, this hippodrome covers all the points of $\text{ball}(o', r)$ that are above h. This implies that o maximizes its distance to $\bigcirc U_i$ if $o = u$.

So let $\tau' = \|u - u'\|$. By the above sanity condition, the segment to' and uu' meet at a right angle, and, hence, by similar triangles (see Figure 8), we have

$$\tau' = \frac{\ell' + \rho}{\ell'}\tau = \tau + \rho\frac{\tau}{\ell'} = \tau + \rho\sin\beta = \tau + \rho\frac{r}{\sqrt{\ell^2 + r^2}} = \tau + \rho\frac{r}{\sqrt{1 + r^2}} \leq \tau + \rho r.$$

This implies, by Equation (4.1), that

$$\frac{d\left(o, \bigcirc U_i\right)}{d\left(o, \bigcirc U_{i-1}\right)} \leq \frac{\|u - u'\|}{\|z - o'\|} = \frac{\tau'}{r - \varepsilon} \leq \frac{\tau}{r - \varepsilon} + \frac{\rho r}{r - \varepsilon} \leq \frac{\tau}{r - \varepsilon} + 2\rho \leq 1 - \frac{r^2}{4} + 2\sqrt{r\varepsilon} \leq 1 - \varepsilon^{2/3}$$

if $r \geq 8\varepsilon^{1/3}$. □

Lemma 4.4. *Let P be a set of n points in $\mathbb{R}^d$ with diameter $\Delta = \text{diam}(P)$, and let $\varepsilon > 0$ be a parameter; then one can compute a set $U \subseteq P$, such that*

(i) $d_H(\bigcirc U, \bigcirc P) \leq (8\varepsilon^{1/3} + \varepsilon)\Delta$,
(ii) $m = |U| \leq O(k_{\text{opt}}/\varepsilon^{2/3})$, *where* $k_{\text{opt}} = k_{\text{opt}}(P, \varepsilon)$, *and*
(iii) the running time is $O(nm^2 d/\varepsilon^2)$.

Proof. Recall that in any round before the algorithm terminates $r_i > \delta\Delta = 8\varepsilon^{1/3}\Delta$. Let $P_{\text{opt}} = \text{opt}(P, \varepsilon)$ be any optimal approximating set of size k_{opt}. In the ith iteration of the algorithm, for some point $o_i \in P_{\text{opt}}$, its distance to the convex hull of $\bigcirc U_i$ shrinks by a factor of $1 - \varepsilon^{2/3}$, by Lemma 4.3. Conceptually, we charge round i to o_i. Now note that by Claim 4.2, $d(o_i, \bigcirc U_{i-1}) \geq r_i - \varepsilon\Delta > (\delta - \varepsilon)\Delta \geq \Delta\delta/2$. Therefore, once the distance of an optimal point o to $\bigcirc U_i$ falls below $\Delta\delta/2 = 8\varepsilon^{1/3}\Delta/2$, it cannot be charged again in any future iteration. The initial distance of o to $\bigcirc U_0$ is at most Δ. As such, by Lemma 4.3, an optimal point o can get charged at most k times, where k is the smallest positive integer such that $(1 - \varepsilon^{2/3})^k\Delta \leq 4\varepsilon^{1/3}\Delta$, which holds if $\exp(-k\varepsilon^{2/3}) \leq 4\varepsilon^{1/3}$. Namely, $k = O(\varepsilon^{-2/3}\log 1/\varepsilon)$.

Using the same idea of decreasing values of ε, as done in Lemma 2.7, one can improve this bound to $O(1/\varepsilon^{2/3})$. We omit the easy but tedious details. We conclude that the number of iterations performed by the algorithm is at most $m = O(k_{\text{opt}}/\varepsilon^{2/3})$.

So the distance of all the points of P_{opt} from $\bigcirc U_m$ is at most $\delta\Delta$. Now, consider any point $p \in \bigcirc P$. Let $t = \text{nn}(p, \bigcirc P_{\text{opt}})$, and observe that $\|p - t\| \leq \varepsilon\Delta$. Since $t \in \bigcirc P_{\text{opt}}$, we have that t can be written as a convex combination $t = \sum_{i=1}^{\nu} \alpha_i o_i$, where $\alpha_1, \ldots, \alpha_\nu \geq 0$, $\sum_i \alpha_i = 1$, and $o_1, \ldots, o_\nu \in P_{\text{opt}}$. For $i = 1, \ldots, \nu$, let $o_i' = \text{nn}(o_i, \bigcirc U_m)$, and note that $t' = \sum_i \alpha_i o_i' \in \bigcirc U_m$. Now observe that for all i,

$\|o_i - o_i'\| \le \delta\Delta$. In particular, $(o_i - o_i') \in \text{ball}(0, \delta\Delta)$, and hence $\sum_i \alpha_i(o_i - o_i') \in \text{ball}(0, \delta\Delta)$. Therefore, $d(p, \bigcirc U_m) \le \|p - t'\| \le \|p - t\| + \|t - t'\| \le \varepsilon\Delta + \|\sum_i \alpha_i(o_i - o_i')\| \le (\varepsilon + \delta)\Delta$. We conclude that $d_H(\bigcirc U_m, \bigcirc P) \le (\varepsilon + \delta)\Delta$.

As for the running time, at each iteration, the algorithm computes the point in P furthest away from $\bigcirc U_i$. The analysis above assumes these queries are done exactly, which is expensive. However, by Lemma 2.7, one can use faster $\varepsilon\Delta$-approximate queries. Specifically, in each iteration, for each point $p \in P$, use Lemma 2.7 to compute an additive $\varepsilon\Delta$-approximation to its distance to $\bigcirc U_i$, and then select the point in P with the largest returned approximate distance. It is easy to verify that this does not change the correctness of the algorithm. Specifically, the point t_i chosen in the ith round may now be $\varepsilon\Delta$ closer to the current convex hull than the furthest point, and so in the analysis of Lemma 4.3, o_i may lie as much as $\varepsilon\Delta$ above t_i. In particular, the length of τ does not change; however, now ρ is only bounded by $2\sqrt{r\varepsilon}$ instead of $\sqrt{2r_i\varepsilon}$, and this constant factor difference only slightly degrades the constant in front of $\varepsilon^{2/3}$ in the lemma statement. The other effect is that when the algorithm stops the distance to the convex hull is bounded by $(8\varepsilon^{1/3} + \varepsilon)\Delta$, and this is accounted for in the above theorem statement.

Now using Lemma 2.7 directly, it takes $O(nmd/\varepsilon^2)$ time per round to find the $\varepsilon\Delta$ approximate furthest point, and therefore the total running time is $O(nm^2d/\varepsilon^2)$. □

4.2.1 Improving the Running Time Further. The running time of the algorithm of Lemma 4.4 can be improved further, but it requires some care. Let $L_{i-1} = \text{span}(U_{i-1})$ denote the linear subspace spanned by the point set U_{i-1}, with the orthonormal basis $v_1, \ldots v_{i-1}$. For any point $p \in P$, let p'_{i-1} denote its orthogonal projection onto the subspace; that is, $p'_{i-1} = \text{nn}(p, L_{i-1}) = \sum_{j=1}^{i} \langle p, v_j \rangle v_j$, and let $\ell_{i-1}(p) = \|p - p_{i-1}\| = d(p, L_i)$. Observe that for any point $t \in L_{i-1}$ and any point $p \in \mathbb{R}^d$, we have that $\|p - t\| = \sqrt{\|p - p'_{i-1}\|^2 + \|p'_{i-1} - t\|^2}$ by the Pythagorean theorem, where p'_{i-1} is the projection of p to L_{i-1}.

As such, for any point $p \in P$, in the beginning of the ith iteration, the algorithm has the projection and distance of p to L_{i-1}; that is, $p'_{i-1} = (\langle p, v_1 \rangle, \ldots, \langle p, v_{i-1} \rangle)$ and $\ell_{i-1}(p)$. The algorithm also initially computes for each point $p \in P$ its norm $\|p\|^2$. Therefore, given any point $t \in L_{i-1}$, its distance to a point $p \in P$ can be computed in $O(i)$ time (instead of $O(d)$). The algorithm also maintains, for every point $p \in P$, an approximate nearest neighbor $\text{nn}_{i-1}(p) \in \bigcirc U_{i-1}$; that is,

$$d(p, \bigcirc U_{i-1}) \le \|p - \text{nn}_{i-1}(p)\| \le d(p, \bigcirc U_{i-1}) + \varepsilon\Delta,$$

where $\Delta = \text{diam}(P)$. Naturally, the algorithm also maintains the distance $d_{i-1}(p) = \|p - \text{nn}_{i-1}(p)\|$.

Now the algorithm does the following in the ith iteration:

(A) Computes, in $O(n)$ time, the point $p \in P$ that maximizes $d_{i-1}(p)$.
(B) Let p'_{i-1} be the projection of p to L_{i-1}. Computes, in $O(d)$ time, the new vector for the basis of L_i; that is, $v_i = (p - p'_{i-1})/\|p - p'_{i-1}\|$. Now $v_1, \ldots, v_i$ is an orthonormal basis of the linear space L_i.
(C) For every point $p \in P$, update its projection p'_{i-1} into L_{i-1} into the projection of p into L_i by computing $\langle p, v_i \rangle$. Also, update $\ell_i(p) = \sqrt{\ell_{i-1}(p)^2 - \langle p, v_i \rangle^2}$.
(D) Let P' denote the projected points of P into L_i. For every $p \in P$, we need to update $\text{nn}_{i-1}(p)$ to $\text{nn}_i(p)$ (and the associated distance). To this end, the algorithm of Lemma 2.7 is called on p'_i and U_i (all lying in the subspace L_i that is of dimension i). Importantly, the algorithm of Lemma 2.7 is being warm-started with the point $\text{nn}_{i-1}(p)$. Let $\#_i(p)$ be the number of iterations performed inside the algorithm of Lemma 2.7 to update the nearest-neighbor

to p. Observe that the running time for p is $O(\#_i(p)i^2)$, since $i = |U_i|$, the points lie in an i-dimensional space, and, as such, every iteration of the algorithm of Lemma 2.7 takes $O(i^2)$ time.

Lemma 4.5. *For $m = O(k_{\text{opt}}/\varepsilon^{2/3})$, the running time of the above algorithm is $O(nm(d + m/\varepsilon^2 + m^2))$.*

Proof. The algorithm performs $m = O(k_{\text{opt}}/\varepsilon^{2/3})$ iterations, and this bounds the dimension of the output subspace. Every iteration of the algorithm takes $O(nd)$ time, except for the last portion of updating the approximate nearest point for all the points of P (i.e., (D)). The key observation is that $\sum_i (\#_i(p) - 1) = O(1/\varepsilon^2)$, since if the algorithm of Lemma 2.7 runs $\alpha = \#_i(p) > 1$ iterations, then the distance of p to the convex-hull shrinks by a factor of $(1 - \varepsilon^2/2)^\alpha$. Arguing as in the proof of Lemma 2.7, this can happen $O(1/\varepsilon^2)$ times before p is in distance at most $\varepsilon\Delta$ from the convex-hull and can no longer be updated. As such, for a single point $p \in P$, the operations in (D) takes overall $\sum_{i=1}^{m} O(i^2(\#_i(p) - 1)) = O(m^2(m + 1/\varepsilon^2))$ time. This implies that the overall running time of the algorithm is $O(n(dm + m^2/\varepsilon^2 + m^3))$. □

4.2.2 The Result.

Theorem 4.6. *Let P be a set of n points in $\mathbb{R}^d$ with diameter $\Delta = \operatorname{diam}(P)$, and let $\varepsilon > 0$ be a parameter; then one can compute a set $U \subseteq P$, such that*

(i) $d_H(\bigcirc U, \bigcirc P) \le (8\varepsilon^{1/3} + \varepsilon)\Delta$, and
(ii) $|U| \le O(k_{\text{opt}}/\varepsilon^{2/3})$, where $k_{\text{opt}} = k_{\text{opt}}(P, \varepsilon)$.

The running time of the algorithm is $O(nm(d + m/\varepsilon^2 + m^2))$ for $m = O(k_{\text{opt}}/\varepsilon^{2/3})$. (Here the constants hidden in the O are independent of the dimension.)

Remark 4.7.

(A) The constants hidden in the O notation used of Theorem 4.6 are independent of the dimension. In comparison to the other algorithms in this article, the approximation quality is slightly worse. However, the advantage is a drastic improvement in the size of the approximation.
(B) The running time of the algorithm of Theorem 4.6 can be further improved, by keeping track for each point $p \in P$, and each point $t \in U_i$, the distance of t from the hyperplane (in L_i) that determines whether the approximate nearest neighbor to p needs to be recomputed. By careful implementation, this can be done in the ith iteration in $O(in)$ time (updating $O(in)$ such numbers in this iteration). This improves the running time to $O(nm(d + m/\varepsilon^2))$. Motivated by our laziness, we omit the messy details.

Remark 4.8. Note that the algorithm is a simple iterative process, which is oblivious to the value of the diameter $\Delta = \operatorname{diam}(P)$ and does not use it directly anywhere. Nevertheless, after $O(k_{\text{opt}}/\varepsilon^{2/3})$ iterations, the solution is an $(8\varepsilon^{1/3} + \varepsilon)\Delta$-approximation to the convex hull. In practice, one may not know the value of k_{opt}, and so this value cannot be used in a stopping condition. However, it is easy to get a 2-approximation Δ', such that $\Delta \le \Delta' \le 2\Delta$ by a linear scan of the points. Then, one can use the check $d(t_i, \bigcirc U_i) = d_H(\bigcirc P, \bigcirc U_i) \le (8\varepsilon^{1/3} + \varepsilon)\Delta'/2$ as a stopping condition, where U_i is the current approximation.

ACKNOWLEDGMENTS

We thank the anonymous referees for their useful feedback.

REFERENCES

[1] P. K. Agarwal, S. Har-Peled, and K. Varadarajan. 2005. Geometric approximation via coresets. In *Combinatorial and Computational Geometry*, J. E. Goodman, J. Pach, and E. Welzl (Eds.). Cambridge, New York, NY.

[2] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. 2004. Approximating extent measures of points. *J. Assoc. Comput. Mach.* 51, 4 (2004), 606–635. DOI: https://doi.org/10.1145/1008731.1008736

[3] S. Arora, R. Ge, and A. Moitra. 2014. New algorithms for learning incoherent and overcomplete dictionaries. In *Proceedings of the 27th Annual Conference on Learning Theory (COLT'14)*. Vol. 35. 779–806.

[4] M.-F. Balcan, A. Blum, and S. Vempala. 2015. Efficient representations for lifelong learning and autoencoding. In *Proceedings of the 28th Annual Conference on Learning Theory (COLT'15)*. Vol. 40. 191–210.

[5] S. Barman. 2015. Approximating Nash equilibria and dense bipartite subgraphs via an approximate version of Caratheodory's theorem. In *Proceedings of the 47th Annual ACM Symposium on the Theory of Computing (STOC'15)*. ACM, 361–369. DOI: https://doi.org/10.1145/2746539.2746566

[6] A. Blum, S. Har-Peled, and B. Raichel. 2015. Sparse approximation via generating point sets. *ArXiv e-prints* (July 2015). arxiv:cs.CG/1507.02574

[7] A. Blum, S. Har-Peled, and B. Raichel. 2016. Sparse approximation via generating point sets. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*, Robert Krauthgamer (Ed.). SIAM, 548–557. DOI: https://doi.org/10.1137/1.9781611974331.ch40

[8] G. Van Buskirk, B. Raichel, and N. Ruozzi. 2017. Sparse approximate conic hulls. In *Proceedings of the Annual Conference on Neural Information Processing Systems: Advances in Neural Information Processing Systems 30 (NIPS'17)*. 2531–2541.

[9] K. L. Clarkson. 1993. Algorithms for polytope covering and approximation. In *Proceedings of the 3rd Workshop on Algorithms and Data Structure (WADS'93)*. Lecture Notes in Computer Science, Vol. 709. Springer-Verlag, 246–252.

[10] K. L. Clarkson. 2010. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Trans. Algor.* 6, 4 (2010), 63:1–63:30. DOI: https://doi.org/10.1145/1824777.1824783

[11] T. Gonzalez. 1985. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.* 38 (1985), 293–306.

[12] S. Har-Peled. 2011. *Geometric Approximation Algorithms*. Mathematical Surveys and Monographs, Vol. 173. American Mathematical Society, Boston, MA.

[13] S. Har-Peled, N. Kumar, D. Mount, and B. Raichel. 2015. Space exploration via proximity search. In *Proceedings of the 31st Annual Symposium on Computer Geometry (SoCG'15)*. Vol. 34. 374–389. DOI: https://doi.org/10.4230/LIPIcs.SOCG.2015.374

[14] S. Har-Peled and K. R. Varadarajan. 2004. High-dimensional shape fitting in linear time. *Discrete Comput. Geom.* 32, 2 (2004), 269–288.

[15] D. Haussler and E. Welzl. 1987. ε-nets and simplex range queries. *Discrete Comput. Geom.* 2 (1987), 127–151.

[16] P. M. Long. 2001. Using the pseudo-dimension to analyze approximation algorithms for integer programming. In *Proceedings of the 7th Workshop on Algorithms and Data Structure (WADS'93)*. Lecture Notes in Computer Science, Vol. 2125. 26–37.

[17] V. Mirrokni, R. P. Leme, A. Vladu, and S. C. Wong. 2017. Tight bounds for approximate Carathéodory and beyond. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. 2440–2448.

[18] A. B. J. Novikoff. 1962. On convergence proofs on perceptrons. In *Proceedings of the Symposium on Mathematics and Theoretical Automata*. Vol. 12. 615–622.

[19] D. A. Spielman, H. Wang, and J. Wright. 2012. Exact recovery of sparsely-used dictionaries. In *Proceedings of the 25th Annual Conference on Learning Theory (COLT'12)*. 37.1–37.18.

Received June 2017; revised August 2018; accepted December 2018